FLASH-PELIOHJELMOINTI

Lauri Paakinaho

Opinnäytetyö
Huhtikuu 2011
Tietotekniikka
Ohjelmistotekniikka
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikka, Ohjelmistotekniikka

Tekijä: Lauri Paakinaho
Työn nimi: Flash-peliohjelmointi
Sivumäärä: 36
Valmistumisaika: 28.5.2011
Työn ohjaaja: Tony Torp

Tiivistelmä

Tämän työn tarkoituksena oli toteuttaa kaupalliseen käyttöön soveltuva internet selaimessa toimiva tietokonepeli Flash-tekniikalla. Työssä tutustuttiin myös Flashin historiaan ja ominaisuuksiin sekä käytetyihin työkaluihin.

Flash on Adoben kehittämä multimedia-alusta joka julkaistiin ensimmäisen kerran vuonna 1996. Flash kehitettiin alunperin animaatioeditoriksi, mutta kyseinen teknologia on sittemmin laajentunut monipuoliseksi kehitysalustaksi jolla voidaan tuottaa interaktiivista sisältöä internet sivuille.

Flash tarjoaa erinomaiset työkalut pelikehitykseen. Peliprojektit saadaan Flashin avulla nopeasti liikkeelle Flashin työkalujen ja valmiiden luokkien ansiosta.

Suunnittelu on yksi pelikehityksen tärkeimmistä osa-alueista. Projekti kannattaa pilkkoa suuremmiksi kokaonaisuuksiksi joiden pohjalta voidaan alkaa tarkemmin miettimään pienempiä yksityiskohtia. Hyvän suunnitelman pohjalta on helpompi lähteä ohjelmoimaan peliä.

Ohjelmointi Flashille tapahtuu ActionScript-ohjelmointikielellä. Ohjelmoinnin kannalta tämän projektin tärkeimmät osa-alueet olivat törmäystunnistuksen, animoinnin ja tekoälyn toteutus.

Avainsanat Flash, ActionScript, pelikehitys

TAMK University of Applied Sciences
Computer Science, Software Engineering

Writer:	Lauri Paakinaho
Thesis:	Flash-gameprogramming
Pages:	36
Graduation time:	28.5.2011
Thesis Supervisor:	Tony Torp

Abstract

The purpose of this thesis was to develop a commercial browser game with the Flash development platform and to get acquainted with Flash technology in general. A brief glance is give for the history of Flash and the development tools used in this project.

Flash is a multimedia platform developed by Adobe. Flash was first released in 1996. Flash was originally created as an animation tool but has since then grown in to a diverse development platform that can be used to create interactive components for websites.

Flash provides excellent tools for game development. Game projects can be started rapidly with the tools and classes provided by Flash.

Design is one of the most important aspects of game development. A good way to design a big project is to divide it into larger sections which are then defined in to smaller instances. A good design gives great foundations for the programmers to work with.

Programming in Flash is done with the ActionScript programming language. Collision detection, animation and artificial intelligence were the key aspects of the programming work done for this project.

Keywords Flash, ActionScript, game development

Sisällysluettelo

LYHENTEET JA TERMISTÖ

Flash	Adoben kehittämä multimedia-alusta.
ActionScript	Flashin hallitsemiseen käytetty ohjelmointikieli.
Aikajana	Flash-työkaluista löytyvä ominaisuus jolla voidaan helposti toteuttaa animaatiota.
MovieClip	Flashin valmis luokka jota käytetään yleisesti Flash-pelikehityksessä.
Tile	Neliön muotoinen grafiikkaobjekti joista muodostetaan pelin kentät tile-pohjaisissa peleissä.
Layer	Pelejä varten kehitetty luokka joka toimii piirtoalustana pelin muille olioille.

1. Johdanto

Flash on Adoben kehittämä multimedia-alusta jolla voidaan kehittää pelejä, animaatioita, videoita ja muita interaktiivisia komponentteja internet sivuille. Flashia voidaan käyttää useissa eri tietokonejärjestelmissä jotka tukevat Adobe Flash Playeriä.

Flashilla voidaan esittää vektori- ja rasterigrafiikkaa jota voidaan hallita Flashia varten kehitetyllä ActionScript-ohjelmointikielellä. ActionScriptin on Adoben Flashia varten kehittämä olio-ohjelmointikieli.

Tämän työn tarkoituksena on suunnitella ja toteuttaa kaupalliseen käyttöön soveltuva selainpeli Flash-tekniikalla. Työ on jaettu karkeasti neljään osaa: Flashin perusteisiin, pelin suunnitteluun, pelin toteutukseen ja pelin viimeistelyyn.

Ensimmäisessä osiossa tutustutaan Flashin historiaan ja Flash-pelimarkkinoihin. Osiossa tutustutaan myös työn toteutuksessa käytettyihin työkaluihin ja käydään läpi hieman Flash-pelikehityksen perusteita.

Toisessa osiossa paneudutaan pelisuunnitteluun. Osiossa tarkastellaan kuinka pelin kehitys jaetaan suuremmiksi kokonaisuuksiksi jotka vähitellen pilkotaan pienemmiksi palasiksi. Suunnittelun tarkoituksena on saada aikaseksi selvä kuva siitä mitä halutaan tehdä ja kuinka haluttu tulos saavutetaan.

Kolmas osio keskittyy pelin ohjelmointiin. Siinä käydään läpi pelin toteutus vaihe vaiheelta ja tarkastellaan useita koodiesimerkkejä.

Viimeisessä osiossa keskitytään pelin hiomiseen valmiiksi paketiksi. Osiossa käydään läpi viimeistelyyn tarvittavat toimenpiteet.

2. Adobe Flash

Adobe Flash on Adoben kehittämä multimedia-alusta jolla voidaan kehittää pelejä, animaatioita, videoita ja muita interaktiivisia komponentteja internet sivuille. Flashia voidaan käyttää useissa eri tietokonejärjestelmissä jotka tukevat Adobe Flash Playeriä. Tässä osiossa tutustutaan Flashin historiaan, työkaluihin ja pelikehitykseen.

2.1 Flashin historia

Flash kehitettiin alunperin Macromedian toimesta ja julkaistiin vuonna 1996. Sittemmin Adobe osti Macromedian ja on jatkanut edelleen alustan kehitystä. Flashin alkuperäinen käyttötarkoitus oli luoda käyttäjille helppo tapa tuottaa animaatioita internet sivustoille. Kehitysympäristö tarjosi vain perustason editointityökalut ja frame-pohjaisen aikajanan jonka avulla voitiin hallita animaation kulkua. (*Wikipedia: Adobe Flash. Viitattu 13.4.2011*)

Ensimmäinen suuri kehitysaskel tapahtui vuonna 2000, kun Flashin viidenteen versioon lisättiin ActionScript 1.0 ohjelmointikieli. ActionScriptin lisääminen antoi käyttäjille mahdollisuuden manipuloida animaatioelementtejä ja animaatioaikajanaa kooditasolta. ActionScript mahdollisti pelikehityksen alkamisen Flash-alustalla. (*Wikipedia: Adobe Flash. Viitattu 13.4.2011*)

Vuonna 2003 julkaistiin Flashin seitsemäs versio, joka sisälsi ActionScript 2.0 ohjelmointikielen. ActionScript 2.0 toi karkean olio-ohjelmointimallin Flash-kehitysympäristöön. (*Wikipedia: Adobe Flash. Viitattu 13.4.2011*)

Flashin yhdeksäs versio julkaistiin vuonna 2007. Yhdeksännen version mukana tuli ActionScript 3.0 ohjelmointikieli joka vei Flashin täysin olio-pohjaiseen kehitykseen ja mahdollisti kokonaisten Flash-aplikaatioiden kehittämisen pelkän ohjelmoinnin avulla. (*Wikipedia: Adobe Flash. Viitattu 13.4.2011*)

Tällä hetkellä uusin versio Flashista on Flash CS5 joka sisältää muun muassa tuen 3D-pelikehitykselle sekä paketoijan jonka avulla Flash-sovelluksia voidaan kääntää suoraan Applen iOS-laitteille. Paketoija kääntää ActionScript 3.0 koodit suoraan natiiviksi Objective-C koodiksi.

2.2 Flash-pelimarkkinat

Flash-pelimarkkinat ovat olleet huomattavassa kasvussa viime vuosien aikana. Internet on täynnä erilaisia pelisivustoja jotka haalivat parhaat verkosta löytävät pelit sivuilleen. Pelit ovat käyttäjille pääosin ilmaisia. Pelisivustojen tulos perustuu sivustoilla esitettäviin mainoksiin. Mitä suurempi kävijämäärä sivustolla on, sitä enemmän mainokset tuottavat rahaa. Sivustojen ylläpitäjille pelit ovat erinomainen tapa houkutella käyttäjiä sivustoilleen. Tästä johtuen pelisivustojen ylläpitäjät ovat kiinnostuneita hankkimaan oikeidet laadukkaisiin Flash-peleihin.

Flash-pelien myynnissä käytetään pääsääntöisesti niin sanottua sponsorointimallia. Tässä mallissa pelikehittäjä tuottaa pelinsä alusta loppuun omalla budjetillaan jonka jälkeen kehittäjä alkaa kaupittelemaan peliään eri pelisivustoilla. Mikäli taloudelliseen yhteisymmärrykseen päästään, siirrytään pelin brändäämiseen. Peliin sijoitetaan useita sponsorin logoja ja linkkejä joita klikkaamalla pelaaja siirretään sponsorin ylläpitämälle pelisivustolle missä pelaajalla on mahdollisuus peleta muita pelejä.

Sponsori levittää brändättyä peliä kaikille mahdollisille pelisivustoilla missä pelaajat voivat peliä pelata. Erilaisia pelisivustoja on tarjolla verkossa niin suuria määriä, että levittämällä peliä oikein, voi peli saada kymmeniä miljoonia palaajia näin ollen esittäen sponsorin mainokset kymmenille miljoonille ihmisille.

Flash-markkinoilla on myös muitakin kaupallisia vaihtoehtoja sponsorointimallin lisäksi. Kehittäjä voi asettaa peliinsä internet mainontaa tarjoavien yritysten kuten Googlen mainoksia peliinsä. Mainoksia esitetään pelin aikana ja kehittäjä saa osan mainosten tuottamasta tulosta. Pelissä voidaan myös hyödyntää niin sanottuja mikromaksuja. Ne ovat pelin lisäsisältöä jonka pelaajat voivat avata maksamalle

kehittäjälle tietyt summan. Omia mainoksia tai mikromaksuja hyödyntäessään kehittäjä on kuitenki itse vastuussa pelin levittämisestä mikä saattaa olla haastavaa, mikäli kehittäjällä ei ole vahvaa tietämystä erilaisista verkoista joidenka avulla peliä voidaan levittää suurille määrille pelisivustoja.

Myös Facebook tarjoaa hyvän mahdollisuuden kaupalliseen Flash-pelikehitykseen. Facebookin valtavien käyttäjämäärien ansiosta hyvät pelit voivat Facebook-mainonnan avulla tienata suuria summia. Facebook-kehityksessä suuret yritykset ovat tosin jo ottaneet johtoaseman joten itsenäiselle pelinkehittäjälle voi olla äärimmäisen vaikeaa kilpailla suurten yritysten resurssien kanssa.

Vaikka valtaosa Flash-peleistä onkin selaimessa toimivia pelejä, se ei tarkoita sitä etteikö Flashilla voisi tehdä muunkinlaisia pelejä. Flash tarjoaa mahdollisuuden paketoida pelin Air-aplikaatioksi joka voidaan asentaa käyttäjän käyttöjärjestelmälle kuten mikä tahansa muukin ohjelma. Tämä mahdollistaa myös melko suurtenkin pelien tekemisen Flashilla, koska Air-aplikaatiot asentuvat käyttäjän koneelle eikä niitä tarvitse ladata jokaisella pelikerralla selaimen avulla.

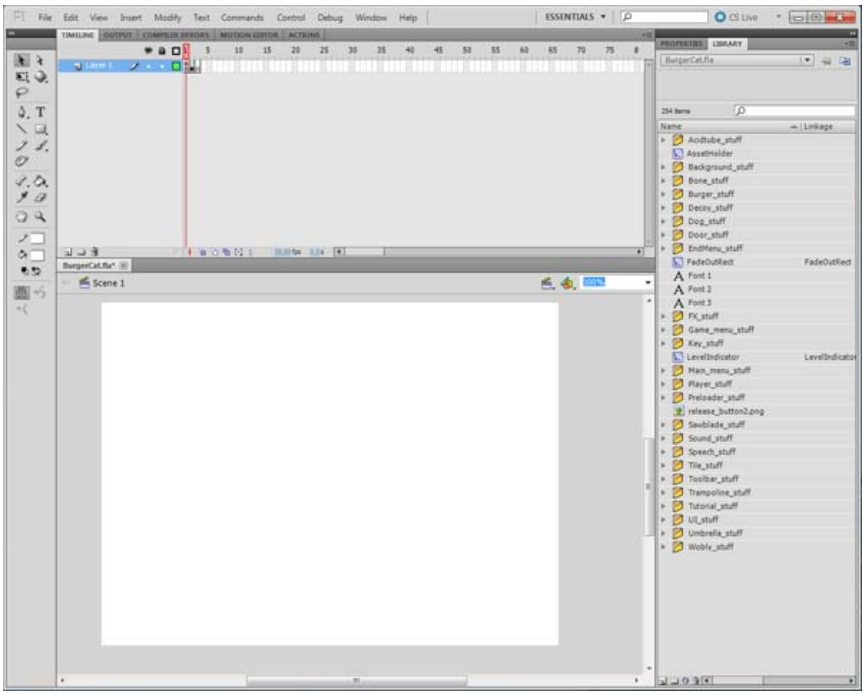
Uusia Flash-pelejä ilmestyy kymmenittäin joka päivä. Kuitenkin vain äärimmäisen pieni osa näistä peleistä on ammattilaisten tekemiä laadukkaita pelejä. Tämä antaa päättäväisille kehittäjille hyvän mahdollisuuden erottua heikkolaatuisesta massasta ja luoda uraa Flash-pelimarkkinoilla. Kaiken kaikkiaan Flash tarjoaa erittäin laajat työkalut monipuoliseen pelikehitykseen.

2.3 Työkalut

Flash-kehitykseen on saatavilla useita eri työkaluja kuten esimerkiksi open source pohjainen FlashDevelop. Tässä työssä kehitysympäristönä toimii Adobe Flash CS5. Eri kehitysympäristöjen välillä esiintyy suuriakin eroja kehitystavoissa. Tästä johtuen kaikkia tässä työssä sovellettuja kehitysmetodeja ei voida soveltaa kaikissa kehitysympäristöissä.

FlashDevelopista poiketen Adobe Flash CS5 mahdollistaa grafiikka- ja audioresurssien tuomisen suoraan kehitysympäristöön ja niiden manipuloimisen kehitysympäristön omilla työkaluilla. Tästä johtuen Adoben työkaluilla kehittäminen on jokseenkin helpompaa kuin monilla vaihtoehtoisilla kehitysympäristöillä, joissa kaikki resurssienhallinta täytyy hoitaa koodin tasolla.

Adobe Flash CS5 tarjoaa myös aikaja-animoinnin mikä helpottaa huomattavasti graafisten elementtien animoimista. Adobe Flash CS5 kehitysympäristö esitettynä kuviossa 1.



Kuvio 1: Adobe Flash CS 5 kehitysympäristö

2.4 Flash-pelikehityksen perusteet

Jotta Flashilla voidaan kehittää pelejä täytyy ymmärtää joitakin Flashin perustason ominaisuuksia. Flashissa grafiikan esittämisen pohjana on Stage-luokka. Stage-olio luodaan automaattisesti, luotaessa uusi Flash-projekti. Perusmuodossaan Stage on vain tyhjä valkoinen alue jonka koko voidaan määrittää kehitysympäristön parametreja säätämällä. Stagelle määrätty koko määrittää Flash-pelin graafisen näkymän koon.

Stagelle voidaan lisätä grafiikkaa Flashin tarjoamien DisplayObjectContainer-luokasta periytyvien olioiden avulla. Grafiikan esittämiseen on tarjolla useita erilaisia luokkia jotka kaikki periytyvät DisplayObjectContainer-luokasta. DisplayObjectContainer on olio johon voidaan liittää muita graafisten luokkien olioita ja hallita niitä. Kun DisplayObjectContainer lisätään Stagen lapseksi, se tulee näkyviin ruudulle. DisplayObjectContainer on siis ikään kuin kerros jonka päälle voidaan piirtää mitä tahansa grafiikkaa.

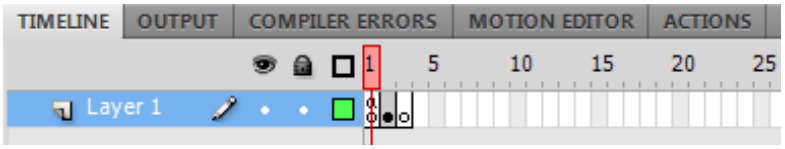
Kaikki ruudulla näkyvät graafiset oliot lisätään Flashin ylläpitämään Display Listiin. Kun Diplay Listiin lisätään olio, se ilmestyy ruudulla näkyvistä olioita päällimmäiseksi. Display Listin avulla voidaan määrittää olioiden syvyyssuhteita muihin oliohin rinnastettuna. Display Listin voidaan helposti siis näyttää tai piilottaa olioita näkyvistä. Olioden poistaminen Display Lististä ei kuitenkaan tuhoa oliota vaan ainoastaan piilottaa sen näkyvistä. Kaiken olioon liittyvän koodin suorittamista jatketaa huolimatta siitä onko se näkyvissä vai ei.

Parhaiten pelikehitykseen soveltuva DisplayObjectContainerista perityvä luokka on MovieClip-luokka. MovieClip-luokkaan kuuluu useita valmiita muuttujia ja metodeja kuten esimerkiksi olion sijainti ruudulla, skaalaus ja alpha-kanava. MovieClip-luokka eroaa muista graafisista luokista sillä, että jokaisella MovieClip-luokan oliolla on oma aikajana jota voidaan käyttää hyväksi animoimisessa.

Adobe Flash CS5-kehitysympäristössä on mahdollista luoda uusia DisplayObjectContainer-luokasta tai sen perillisistä periytyviä luokkia graafisen käyttöliittymän avulla. Graafisella käyttöliittymällä määritetään vain luokan graafisia ominaisuuksia. Luokaa luotaessa määritetään luokan nimi ja sitä hallinnoivan ActionScript-kooditiedoston nimi. ActionScript-tiedosto sisältää luokan määrittelyn ja toteutuksen. Helpoin tapa pelikehityksen kannalta on luoda uusia luokkia jotka periytyvät MovieClip-luokasta, koska näin saadaan käyttöön kaikki DisplayObjectContainerin sisältämät ominaisuudet sekä MovieClip-luokalle ominainen aikajana käyttöön.

Aikajana(kuvio 2) koostuu frameista. Kehitysympäristön avulla voidaan määrittää framejen määrä ja jokaiseen frameen voidaan tuoda grafiikkaa. Esimerkiksi jos

pelihahmolle halutaan luoda kävelyanimaatio luodaan kehitysympäristössä uusi MovieClip-luokasta perityvä luokka. Kehitysympäristössä määritetään framejen määrä ja tuodaan niihin grafiikat. Luokan määrittelevästä ActionScript-tiedostosta voidaan tämän jälkeen koodin tasolla määrittää mikä frame esitetään milläkin hetkellä ja näin saadaan aikaiseksi animaatiota.



Kuvio 2: MovieClip-olion aikajana

Jotta Flash-aplikaatiota voidaan alkaa hallitsemaan ohjelmoinnin avulla, täytyy kehitysympäristöön määrittään niin sanottu Document Class. Se on luokka josta luodaan olio automaattisesti aina aplikaation suorituksen alkaessa. Se toimii siis ohjelman koodin lähtöpisteenä ikään kuin Main-luokka useissa muissa ohjelmointikielissä. Document Class-luokaksi määritellyn luokan avulla voidaan sitten alkaa koodin tasolla hallitsemaan mitä olioita luodaan ja esitetään ruudulla. Näistä peruslähtökohdista voidaan lähteä toteuttamaan varsinaista peliä.

3. Pelisuunnittelu

Suunnittelulla on suuri rooli pelikehityksessä. Hyvän suunnittelun avulla voidaan arvioida pelin kehityskustannuksia ja sen toteuttamiseen vaadittavaa aikaa. Hyvä suunnittelu vähentää kehitykseen kuluvaa aikaa huomattavasti. Tämän osion tarkoitus on luoda katsaus tässä projektissa toteutetun pelin eri suunnitteluvaiheisiin.

3.1 Peli-idea

Tässä työssä toteutettiin puzzle- eli ongelmanratkaisupeli nimeltä Burger Cat. Pelin päähahmona toimii kissa, jota tekoäly ohjaa. Pelin ideana on rakentaa kissalle turvallinen reitti kentän läpi hampurilaisen luo. Valmiista pelistä otettu kuvakaappaus esitettynä kuviossa 3.

Kuvio 3: Kuvakaappaus pelitilanteesta

Jokaisessa kentässä pelaajalla on käytössään rajoitettu määrä erilaisia työkaluja. Käytettävissä olevat työkalut sijaitsevat peliruudun yläreunassa olevassa työkalupaneelissa. Jokaisen työkaluikonin yhteydessä on numero joka kertoo kuinka monta kyseistä työkalua pelaajalla on käytössä.

Työkaluja käytetään klikkaamalla työkaluikonia hiirellä ja sen jälkeen klikkaamalla haluttua kohtaa kentässä. Työkalun valitsemisen jälkeen hiiren osoittimeen liitetään grafiikka joka kuvastaa voiko kyseistä työkalua käyttää johonkin tiettyyn kohtaan kentässä. Klikattaessa kenttää valittu objekti putoaa kentällä haluttuun kohtaan.

Työkalujen avulla pelaajan tulee auttaa kissaa välttämään kentissä vaanivat vaarat. Pelissä on useita vihollisia kuten koiria, sahanteriä ja pohjattomia kuiluja. Mikäli kissa osuu johonkin viholliseen tai putoaa kuiluun, kenttä täytyy aloittaa alusta.

Jokaiseseen kenttää sijoitetaan hampurilainen joka toimii pelin maalina. Kun pelaaja saa ohjattua kissan hampurilaisen luo, voi pelaaja siirtyä seuraavaan kenttään.

3.2 Pelin ominaisuudet

Peliin haluttiin toteuttaa laaja joukko erilaisia ominaisuuksia. Suurimman haasteen tarjosi erilaisten työkalujen suunnittelu. Lopulta peliin päädyttiin toteuttamaan seitsemän erilaista työkalua:

1. Taikasauva: taikasauvan avulla pelaaja voi listä pelialueella uusia maa-alueita joita pitkin kissa voi kävellä.
2. Hakku: hakulla pelaaja voi tuhota pelialueella olevia maa-alueita näin ollen raivaten kissalle uusia kulkuväyliä.
3. Trampoliini: trampoliinin avulla kissa voidaan pompauttaa korkeiden esteiden yli.
4. Kumihiiri: pelaaja voi käyttää kumihiirtä kissan houkuttelemiseen. Toisin sanottuna pelaajan sijoittaessa hiiren samalle tasolle kissan kanssa, se vetää kissan huomion puoleensa eli kääntää kissan kohti hiirtä.

5. Puruluu: puruluu toimii vastaavalla tavalla kuin kumihiiri, mutta se tehoaa vain koiriin. Pelissä vihollishahmoina toimivia koiria voidaan häiritä puruluun avulla.
6. Sateenvarjo: sateenvarjon avulla pelaaja voi suojata kissaa happopisaroilta joita esiintyy joissakin pelin kentissä. Sateenvarjo kestää osuman kahdesta happopisarasta.
7. Dynamiitti: dynamiitilla voidaan rikkoa joissakin kentissä esiityviä kiviröykkiöitä.

Peliin toteutettiin myös useita erilaisia esteitä ja vihollisia jotka ovat kissalle vaaraksi:

1. Sahanterä: paikallaan oleva pyörivä sahanterä johon kissa ei saa osua.
2. Koira: koira partioi tiettyä aluetta määrättyjen kriteerien mukaisesti. Mikäli koira osuu seinään tai havaitsee edessään kuopan, se vaihtaa suuntaa. Koira liikkuu kaksinkertaisella nopeudella kissaan verrattuna.
3. Happoputki: happoputkesta putoaa tasaiseen tahtiin happopisaroita jotka ovat kissalle kuolettavia. Pisarat putoavat putken päästä ja tuhoutuvat osuessaan maahan.

Pelialue haluttiin rakentaa erilaisista palasista joihin pelaaja voi vaikuttaa eri tavoille eri työkalujen avulla. Pelialue päädyttiin toteuttamaan seuraavista palasista:

1. Maapala: tavallisin maa-alue joka kuvastaa multaa ja nurmikkoa. Tähän palaan voidaan vaikuttaa hakun ja dynamiitin avulla.
2. Kiviröykkiö: toimii samalla tavalla kuin tavallinenkin maapala, mutta tähän palaan voidaan vaikuttaa ainoastaan dynamiitin avulla.
3. Teräs seinä: tähän palaan ei voida vaikuttaa millään pelistä löytyvillä työkaluilla. Pelaajan on löydettävä reitti kyseisen palan ympäri.
4. Ovi: pelin kentistä voi löytyä kultainen, hopeinen ja pronssinen ovi. Kentässä voi esiintyä yksi tai useampi ovi muttei kuitenkaan useampaa saman väristä ovea. Avatakseen ovet pelaajan tulee ensin ohjata kissa kentästä löytyvän avaimen luokse. Avainten värejä on kolme ja ne vastaavat ovien värejä. Pomittuaan avaimen kentästä, voi kissa kulkea oven läpi.

Peliin toteutettiin myös useita erilaisia valikoita joidenka avulla pelaaja voi siirtyä pelissä eri tilasta toiseen:

1. Päävalikko: valikko jossa esitetään pelin logo, aloituspainike, ”lisää pelejä”-painike ja linkki pelin kehittäjän internet sivuille. Päävalikkoon tuli myös sisällyttää mute-painike jolla voidaan hiljentää pelin äänet. Aloituspainiketta painettaessa pelaaja siirretään kenttävalikkoon.
2. Kenttävalikko: tässä valikossa esitetään 40 painiketta jotka edustavat jokaista pelin kenttää. Jokaisessa painikkeessa tulee ilmetä kentän numero ja onko kenttä lukittu. Uudet kentät aukeavat pelaajan pelatessa peliä. Painettaessa kenttäpainiketta joka ei ole lukittu, pelaaja siirretään peliin pelaajan valitsemaan kenttään.
3. Pelivalikko: pelin aikana näkyvässä työkalupaneelissa on myös valikko jossa sijaitsee useita painikkeita. Valikosta löytyy seuraavat painikkeet:

    a. Reset-painike jonka avulla pelaaja voi aloittaa kentän uudelleen mikäli kentän läpäisy muuttuu mahdottomaksi.
    b. Mute-painike jolla voidaan hiljentää pelin äänet.
    c. Pause-painike jolla voidaan pysäyttää pelitilanne.
    d. Quit-painike jota painamalla pelaaja viedään takaisin päävalikkoon.

Peliin haluttiin edellämainittujen ominaisuuksien lisäksi lisätä myös jonkinlainen onnitteluruutu, joka esitetään, kun pelaaja läpäisee viimeisen kentän. Loppuruudun tulisi pitää sisällään painike jolla pelaaja voi siirtyä takaisin päävalikkoon ja painike jolla pelaaja voi siirtyä tulevan sponsorin sivuille pelaamaan lisää pelejä.

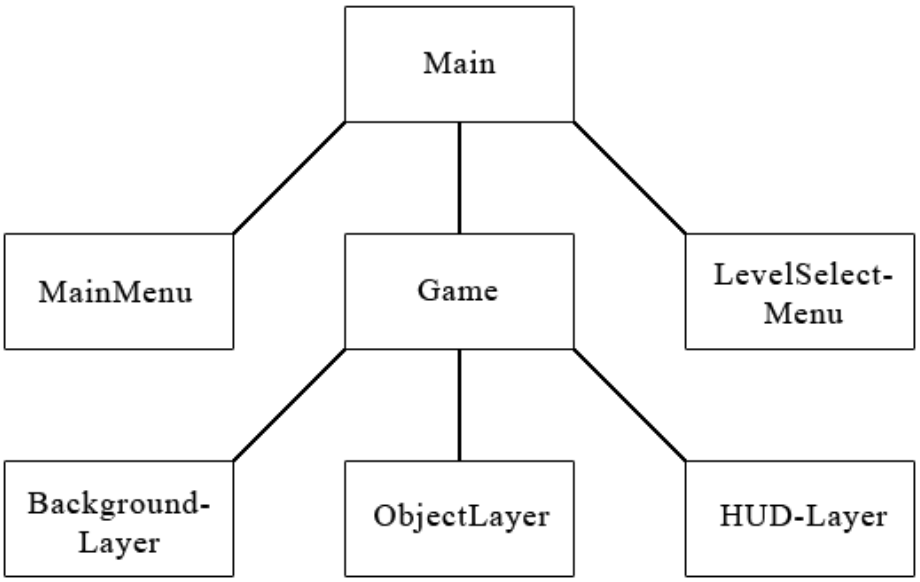
Pelitilanteen tallentaminen päätettiin toteuttaa Local Shared Objectien avulla. Local Shared Object(LSO) on Flashin oma tapa tallentaa tietoa käyttäjän koneelle. LSO:n ainoa huono puoli on, että peliä täytyy pelata samalla pelisivustolla jotta tallennukset toimisivat.

Pelin äänet täytyy myös voida hiljentää missä pelitilanteessa tahansa, koska useat ihmiset pelaavat Flash-pelejä toimistoissa ja kouluissa missä äänistä voisi olla pelaajalle haittaa.

3.3 Pelirungon suunnittelu

Pelirungon suunnittelun alkuvaiheessa kannattaa runkoa lähteä tarkastelemaan mahdollisimman korkealta tasolta ja vähitellen siirtyä suunnittelemaan rungon pienenpiä osia. Suunnittelun alkuvaiheessa tulee miettiä miten suuremmat kokonaisuudet toimivat, minkälaisia luokkia niiden toteuttamiseen tarvitaan ja miten kyseiset luokat sidotaan yhteen suuremmaksi kokonaisuudeksi.

Tässä projektissa peli jaettiin kolmeen suurempaan kokonaisuuteen joita hallinnoi yksi pääluokka. Pelin dokumenttiluokaksi, eli koodin suorituksen alkupisteeksi päätettiin toteuttaa luokka nimeltä Main. Main hallinnoi pelirungon kolmea suurinta luokkaa jotka muodostavat koko pelin kolme tärkeintä osaa. Kuvaus alustavasta pelirungosta kuviossa 4.



Kuvio 4: Pelirunko jaettuna suurempiin kokonaisuuksiin

Mainin alaisuuteen alistettiin kolme luokka: MainMenu, LevelSelectMenu ja Game. MainMenu-luokka toimii pelin päävalikkona joka on ensimmäinen tila johon pelaaja pääsee vaikuttamaan pelin käynnistyttyä. MainMenusta Main-luokka siirtää pelaajan LevelSelectMenuun josta käsin pelaaja valitsee kentän. LevelSelectMenusta pelaaja

siirretään suurimpaan ja tärkeimpään kokonaisuuteen, Game-luokkaan joka toimii pelin varsinaisen pelimoottorin lähtökohta ja on pelin kannalta yksi tärkeimmistä ja monimutkaisimmista luokista.

Game-luokan tehtävänä on luoda kaikki oliot joita tarvitaan jokaisessa pelin kentässä. Sen tulee pystyä lataamaan uusia karttoja karttatiedoston pohjalta, luoda kartoissa määritellyt oliot, piirtää oliot ruudulle ja myöhemmin huolehtia kentän olioiden tuhoamisesta ja uuden kentän lataamisesta.

Koska pelissä on tarve piirtää grafiikkaa useisiin eri kerroksiin kuten pelin taustakuvat, peliolioiden grafiikat ja niiden päälle piirrettävät käyttöliittymä grafiikat, päädyttiin toteuttamaan yksinkertainen Layer-luokka, joka toimii eräänlaisena piirtoalustana joidenka päällekkäisyyksiä on helppo säädellä. Layer-luokan avulla grafiikoiden syvyyksien määrittäminen helpottuu huomattavasti, koska kyseisen luokan avulla pelimoottorin eri osat voidaan helposti piirtää omiin kerroksiinsa eikä tarvitse huolehtia virheellisten päällekkäisykksien syntymisestä.

Pelin kentän päädyttiin toteuttamaan niin sanotulla tile-pohjaisella tyylillä. Tile-pohjaisuus tarkoittaa sitä, että kaikki pelin maastografiikat toteutetaan määrätyn kokoisina neliönä eli tileinä. Tässä projektissa tilejen kooksi määritettiin 32 kertaa 32 pikseliä. Näillä tileillä sitten piirretään karttaeditorin avustuksella kuva, jonka karttaeditori pilkkoo kaksiulotteiseksi taulukoksi, jonka jokaiseen soluun kirjoitetaan numeroarvo joka edustaa jotakin tiettyä tileä. Taulukko tallennetaan erilliseen kartta tiedostoon jossa sijaitsee kaikki pelin kartat.

Game-luokka lukee karttatiedostosta halutun kentän taulukon ja luo sen pohjalta oliota kentälle. Myös pelihahmot luodaan ja niiden sijainti pelialueella määritetään tile-pohjaisen menetelmän avulla.

Tärkein ominaisuus joka tarvitaan pelimoottorin toimimiseksi on törmäystunnistus. Törmäystunnistus jopa kaksiulotteisissa peleissä on kohtuullisen monimutkainen toteuttaa. Törmäyksien havaitseminen on helppoa, mutta niihin reagoiminen oikealla tavalla on haastavaa. Pelimoottoriin päätettiin toteuttaa perustason törmäystunnistus joka toteutettaisiin jokaiselle sitä tarvitsevalle luokalle erikseen, koska kaikkien

törmäystunnistusta tarvitsevien olioiden käyttäytyminen pelialueella vaihtelee oliosta riippuen.

Suunnittelun viimeiseksi suureksi kokonaisuudeksi jäi käyttöliittymän suunnittelu. Edellä läpikäytyjen ominaisuuksien avulla peli tulisi olemaan toimivassa tilassa, mutta pelaaja ei voisi millään tavalla vaikuttaa pelin kulkuun. Käyttöliittymäksi suunniteltiin työkalupalkki(kuvio 5), joka sijaitsee peliruudun ylälaidassa. Työkalupalkki liitettäisiin Game-luokan käyttöliittymä kerrokseen, jolloin kaikki siihen liitettävät oliot olisivat aina ruudulla päällimäisenä.

Kuvio 5: Pelin työkalupalkki

Työkalupalkkiin ladattaisiin ToolButton-luokan olioita erillisen ToolInfo-luokan perusteella, joka tulisi sisältämään taulukoita jotka kertovat jokaisen kentän työkalujen lukumäärät. Niiiden pohjalta luodaan ToolButton-oliot joita klikkaamalla pelaaja voi valita työkalunsa.

Työkalun valinnan jälkeen pelaaja vie työkalun pelialueella klikkaamalla pelialuetta. Tämän jälkeen Game-luokka luo kyseisen olion.

Näiden suunnittelukonseptien pohjalta lähdettiin ohjelmoimaan pelin runkoa ja moottoria.

4. Pelin toteutus

Tässä osiossa tarkastellaan Flash-pelikehitystä ohjelmoinnin kannalta. Osiossa tutustutaan tässä työssä toteutetun pelin ohjelmoinnissa esiintyneisiin haasteisiin ja minkälaisten ratkaisujen avulla niistä selvittiin.

4.1 Pelimoottori

Yksi Flash-kehityksen parhaita puolia on se, että erilaisia pelikonsepteja voidaan testata hyvinkin nopeasti. Tässä projektissa luotiin alustavasti vain pieni Main-luokka jonka tehtävänä oli ainoastaan luoda Game-luokan olio.

Main-luokan luomisen jälkeen lähdettiin suoraan kehittämään varsinaista pelimoottoria eli luokkia jotka toteuttavat pelitapahtumat. Minimissään pelimoottoriin tarvittiin seuraavat luokat: Game, Layer, TileInfo, Tile, MapDimensions, Background ja Player. Näiden luokkien avulla saadaan luotua kenttä ja pelin päähahmo. Minkäänlaista interaktiota pelaajan ja pelin välillä ei vielä pelkästään näiden luokkien totauttemisen jälkeen ole, mutta nämä luokat ovat tärkeimmät pelin fysiikan kannalta.

Moottorin toteutus lähtee liikkeelle Game-luokasta jonka vastuulla on kaikkien muiden peliolioiden luominen. Ensimmäinen Game-luokan tehtävä on luoda MapHolder- , TileInfo- ja MapDimensios-luokat. Ne ovat hyin yksinkertaisia luokkia jotka pitävät sisällään karttat, karttojen koot ja eri tileille määritellyt ominaisuudet.

Seuraavaksi Game luo kolme instanssia Layer-luokasta: bgLayer, objectLayer ja hudLayer. BgLayeriin liitetään kaikki pelin taustoihin liittyvä materiaali, objectLayeriin kaikki pelitilanteeseen vaikuttavat pelioliot kuten pelihahmot ja tilet ja hudLayeriin liitetään pelaajalle tarpeelliset työkalut ja valikot.

Game-luokka luo seuraavaksi Background-luokan olion ja liittää sen bgLayerin lapseksi. Seuraavaksi kutsutaan kentän luomiseen tarvittavia funktioita. LoadMap-

funktion tehtävänä on lukea valitun kentän taulukko MapHolder-oliosta ja piirtää sen perusteella kenttä objectLayeriin.

Kentän luominen tapahtuu kahdessa sisäkkäisessä for-loopissa joidenka pituus määritellään MapDimensions-luokan sisältäminen taulukoiden perusteella. MapDimensions sisältää taulukkoja jotka pitävät sisällään jokaisen kentän leveyden ja korkeuden. For-loopit käyvät läpi kaksiulotteisen karttataulukon solut rivi kerrallaan. For-loopeissa luodaan jokaisen karttataulukon solussa olevan numeroarvon pohjalta Tile-luokan olio joka liitetään objectLayer lapseksi. Mikäli taulukon arvo on nolla, luodaan Tile, joka on näkymätön. Muissa tapauksissa luodaan Tile-luokan olio jonka grafiikka määräytyy taulukon arvon perusteella. Tilelle viedään lisäksi molempien for-looppien sen hetkiset arvot joidenka perusteella Tile-luokan rakennin määrittää oman sijaintinsa kentällä. Kenttien luomiseen käytetyn funktion toimintaa esitettynä listauksessa 1.

```
for(var i:int = 0; i < h; i++)
{
    for(var j:int = 0; j < w; j++)
    {
        if(currentMap[i][j] == 0)
        {
            var tile0:Tile = new Tile(1, 4, j, i);
            tile0.alpha = 0;
            mapObjects.push(tile0);
            objectLayer.addChild(tile0);
        }

        if(currentMap[i][j] > 0)
        {
            var tile:Tile = new Tile(1, currentMap[i][j], j, i);
            mapObjects.push(tile);
            objectLayer.addChild(tile);
        }
    }
}
```

Listaus 1: LoadMap-funktion toimintaa

Tile-luokan rakennin kertoo sille välitetyt sijainin määrittävät numeroarvot Game-luokassa määritellyillä tilejen kokoarvoilla. Tässä pelissä tilejen koko on 32 kertaa 32 pikseliä. Tämän laskutoimituksen seurauksena saadaan kyseisen tilen sijainti pikseleissä.

Kentän luomisen jälkeen Game luo Player-luokan olion ja liittää sen objectLayeriin. Player-luokka edustaa pelin päähahmoa.

Kun kaikki edellä mainitut luokat on toteutettu saadaan kasaan hyvin karkea pelimoottori jonka pohjalta voidaan lähteä luomaan interaktiivisuutta ja lisäsisältöä peliin.

4.2 Törmäystunnistus

Yksi tärkeimmistä pelimoottorin ominaisuuksista on törmäystunnistus tileistä koostuvan kentän ja muiden peliobjektien välillä. Ilman törmäystunnistusta peliolot putoavat kentän läpi. Törmäystunnistus sinällään on varsin helppo toteuttaa, mutta avain toimivaan pelimoottoriin on törmäyksiin reagoiminen oikealla tavalla.

Tile-pohjaisesissa peleissä törmäystunnistus suoritetaan tarkastelemalla peliolioiden sijaintia kyseisen kentän taulukkoon nähden. Pelioliot voivat liikkua x- ja y-akselilla joten jokaiselle liikkuvalle oliolle on määritetty xVel- ja yVel-arvot jotka edustavat liikkeen suuntavektoreita. Törmäystunnistus suoritetaan erikseen x- ja y-akselilla kahdessa eri funktiossa. Ensimmäiseksi suoritetaan sen akselin törmäystunnistus kumman vektorin itseisarvo on suurempi. Molemmissa funktiossa selvitetään ensiksi missä olion ylä- ja alanurkat sijaitsevat siinä vaiheessa, kun olion sijaintiin lisätään x-tai y-vektorin arvo riippuen kumman akselin törmäystunnistusta ollaan suorittamassa. Olion nurkkien sijainnit selvitetään getCorners-funktiossa jonka toimintaa on esitetty listauksessa 2.

```
private function getCorners(xPos:int, yPos:int):void
{
    top = Math.floor((yPos - realH/2) / Game.tileH);
    bottom = Math.floor((yPos + realH/2 - 1) / Game.tileH);
    left = Math.floor((xPos - realW/2) / Game.tileW);
    right = Math.floor((xPos + realW/2 - 1) / Game.tileW);

    upLeft = Game.currentMap[top][left];
    upRight = Game.currentMap[top][right];
    downLeft = Game.currentMap[bottom][left];
    downRight = Game.currentMap[bottom][right];
}
```

Listaus 2: getCorners-funktio joka selvittää olion nurkkien sijainnin

Peliolioiden x- ja y-koordinaatit sijaitsevat olion graafiikoiden keskipisteessä. Grafiikan koko määrittää olion leveyden ja korkeuden. Olion nurkkien sijainti saadaan selville ensin määrittämällä olion reunojen sijainnit karttataulukossa. Esimerkiksi olion yläreunan sijainti saadaan selville vähentämällä y-koordinaatista puolet olion korkeudesta ja jakamalla erotus Game-luokassa määritetyllä tilejen korkeudella. Saatu tulos pyöristetään vielä alaspäin lähimpään kokonaislukuun jolloin saadaan numero joka edustaa yksittäistä riviä karttataulukossa.

Vastaavanlainen laskutoimitus suoritetaan myös vasemmalle-, oikealle- ja alareunalle. Näin saadaan selville kaikkien olion reunojen sijainnit suhteessa kyseisen kentän karttataulukkoon nähden.

Kun reunojen sijainnit on saatu selville, voidaan niiden avulla selvittää kulmien sijainnit. Jokaisella törmäystunnistusta vaativalla pelioliolla on upLeft, upRight, downLeft ja downRight muuttujat joihin asetetaan törmäystunnistuksen yhteydessä karttataulukosta löytyvät arvot. Esimerkiksi solun arvo, jossa olion vasen yläkulma sijaitsee, saadaan karttataulukosta ulos jo selvitettyjen vasemman ja yläreunan avulla.
Jotta törmäystunnistus saataisiin suoritettua nopeammin, törmäystunnistus suoritetaan ainoastaan niillä kulmilla jotka ovat olion liikesuunnan puolella. Eli jos olio liikkuu vasemmalle, tarkastellaan vain vasemmanpuoleisia kulmia. Kulmamuuttujiin sijoitettuja arvoja verrataan TileInfo-luokan tileData-taulukkoon jossa sijaitsee tieto siitä, voiko kyseisen tilen läpi kulkea vai tuleeko törmäykseen reagoida. Jos kyseisen tilen läpi ei voi kulkea, selvitetään kuinka paljon olio tulisi liikkumaan kyseisen tilen sisälle jos sen

sijaintiin lisättäisiin sen hetkisen liikevektorin arvo. Tämän laskutoimituksen avulla saadaan laskettua vektorille arvo joka vie olion tilen viereen, mutta ei sen sisälle. Törmäystunnistukseen tarvittavaa koodia on esitetty listauksessa 3.

```
getCorners(x + xVel, y);

if(xVel < 0)
{
    if(Game.tileInfo.tileData[upLeft] > 1 ||
           Game.tileInfo.tileData[downLeft] > 1)
    {
        tileX = left * Game.tileW + Game.tileW;
        xVel = tileX - (x - realW/2);
        if(Game.tileInfo.tileData[oneUpLeft] == 1 &&
           Game.tileInfo.tileData[upCenter] == 1 && airborne == false)
            state = 3;
        else if(airborne == false)
            state = 1;
    }

}
```

Listaus 3: Törmäyksen tunnistaminen ja siihen reagoiminen

Kun molemmat törmäystunnistusfunktiot on suoritettu, saadaan oliolle x- ja y-liikevektorien arvot jotka pitävät olion kentällä ja estävät sitä kulkemasta seinien läpi. Lopuksi olion sijaintia edustaviin x- ja y-arvoihin lisätään vektorien arvot jolloin olio liikkuu pelialueella.

Peliolioiden keskinäisten törmäysten tunnistaminen on huomattavasti helpompaa. Esimerkiksi Player-luokan olion täytyy selvittää mikäli se törmää vihollisiin tai muihin oliohin joista aiheutuu erinäisiä toimenpiteitä. Jokaiselle pelioliolle on määritetty Rectangle-luokan olio jonka koko ja sijainti määritetään kyseisen olion sijainnin ja koon perusteella.

Flash-tarjoaa Rectangle-luokan oliolle yksinkertaisen Intersects-funktion johon viedään toinen Rectangle-olio. Funktio palauttaa true- tai false-arvon riippuen siitä tapahtuuko suorakulmioiden välillä törmäys. Mikäli törmäys tapahtuu suoritetaan halutut toimenpiteet. Olioiden keskinäiseen törmäystunnistukseen tarvittava koodi on esitettynä listauksessa 4.

```
private function testPlayerCollision():void
{
    if(enemyRect.intersects(Game.player.playerRect) && Game.player.hit
           == false)
    {
        Game.player.takeHit();
    }
}
```

Listaus 4: Törmäystunnistus peliolioiden välillä

4.3 Logiikkaohjelmointi

Jotta pelimoottoriin saataisiin aikaiseksi hieman eloa, täytyy peliolioille ohjelmoida logiikkaa. Logiikka määrittää miten oliot toimivat suhteessa kenttään ja toisiin olioihin. Tässä pelissä olioiden logiikka on jaettu erilaisiin tiloihin. Olion toiminta määräytyy sen kyseisen tilan mukaan.

Jokaiselle graafiselle oliolle luodaan EventListener, joka asetetaan tarkkailemaan enterFrame-eventtiä. Flash kehitystyökaluista voidaan määrittää sovellukselle ruudunpäivitysnopeus. Ruudun päivittyessä ohjelma laukaisee enterFrame-eventin. Jokaisen olion EventListener sidotaan onEnterFrame-funktioon jossa suoritetaan olion logiikkaan, liikkumiseen ja animaatioon liittyvät funktiot. EventListenerien toimintaa on kuvattu listauksessa 5.

```
addEventListener(Event.ENTER_FRAME, onEnterFrame);

private function onEnterFrame(e:Event):void
{
    if(Game.paused == false)
    {
        if(alive && !outOfBounds)
        {
            handleAi();
            handleAnimation();
        }
    }
}
```

Listaus 5: Peliolion logiikan hallintaa EventListener-funktion avulla

Peliolioden logiikka on vahvasti sidottu törmäystunnistukseen. Useimmiten olioiden tilat muuttuvat, kun ne törmäävät seinään tai muihin olioihin. Esimerkiksi kun pelin päähahmona toimiva kissa kävelee päin seinää, tarkastetaan ensin kyseisen seinän korkeus. Mikäli seinä on vain yhden tilen korkuinen, siirtyy kissa kiipeämis tilaan jossa suoritetaan seinän päälle kiipeämiseen vaadittavat liikkumistoimenpiteet. Jos taas seinä on korkeampi, kissa kääntyy ympäri ja jatkaa kävelyään toiseen suuntaan.

Kaikilla pelihahmoina toimivilla luokilla on omat yksilölliset tilansa ja toimintamallinsa, mutta pääsääntöisesti kaikki luokat toimivat samojen sääntöjen alaisuudessa.

Olioiden animoiminen liittyy hyvin läheisesti olioiden logiikkaan. Animointi-funktio tarkistaa missä tilassa olio on tietyllä hetkellä ja suorittaa sen mukaisesti animaatiota. Funktiossa on määritetty jokaiselle tilalle omat framensa, joita funktio pyörittää tilasta riippuen. Hahmon animaatiossa voidaan helposti siirtyä framesta toiseen gotoAndStop-funktiota kutsumalla, johon viedään parametrina vain halutun frame numeroarvo.

MovieClip-luokasta periytyvillä luokilla on käytössään omat animaatioaikajanansa jonne animatioiden eri grafiikat syötetään. GotoAndStop-funktio vain yksinkertaisesti tuo aikajanalta näkyviin valitun framen. Pelihahmojen animointiin liittyvää logiikkaa on kuvattu listauksessa 6.

```
private function handleAnimation():void
{
    //frames 1-12 = walking, 13 = falling
    if((state == 1 || state == 2))
    {
        frame++;
        if(frame > 12)
            frame = 1;
    }

    if(state == 3)
    {
        frame = 13;
    }

    if(xVel > 0)
        scaleX = 1;
    else if(xVel < 0)
        scaleX = -1;

    gotoAndStop(frame);
}
```

Listaus 6: Pelihahmon animaatiofunktio

4.4 Käyttöliittymä

Pelaajan kannalta yksi pelin tärkeimmistä ominaisuuksista on pelin käyttöliittymä. Tähän peliin toteuttiin täysin hiiripohjainen käyttöliittymä. Pelin käyttöliittymä koostuu ruudun yläreunassa sijaitsevasta ToolBar-oliosta ja pelialueen päällä sijaitsevasta ClickGrid-oliosta. ToolBar-oliosta pelaaja valitsee halutun työkalun ja ClickGridiä painamalla kyseinen työkalu asetetaan pelialueelle.

Työkalujen käyttöä hallitsee ToolHandler-luokka. ToolHandler-luokka sisältää ClickGrid- ja ToolPointer-oliot. ToolBar-luokka taas sisältää ToolButton-luokan olioita. ToolButton-olioihin on liitetty EventListenerit jotka tarkkeilevat MouseEvent.CLICK-eventtejä. Pelaajan klikatessa oliota hiiren vasemmalla painikkeella ToolButton-olio viestittää ToolHandler-oliolle mitä työkalua pelaaja haluaa käyttää. ToolButton-luokan toimintaa on kuvattu listauksessa 7.

```
private function click(e:MouseEvent):void
{
    if(amount > 0 && Game.hud.menuLoaded == false && active == true)
    {
        Game.toolBar.clearFilters();
        Game.toolHandler.activate(type);
        this.filters = [new GlowFilter(0xFFFFFF, 1, 4, 4, 2, 2, false,
           false)];
    }
}
```

Listaus 7: ToolButton-olion toimintaa

ToolHandler-puolestaan aktoivoi ToolPointer-olion joka liitetään hiiren osoittimen yhteyteen. ToolPointer-liikkuu peli alueella pelaajan liikuttaessa hiiren osoitinta. ToolPointer määrittää hiiren sijainnin mukaan missä ruudussa hiiren osoitin on suhteessa sen hetkiseen karttataulukkoon. Karttataulukosta tarkistetaan kyseisessä ruudussa sijaitsevat tilen ominaisuudet. Kaikkia työkaluja ei voida käyttää kaikkien tilejen yhteydessä joten ToolPonter ilmaisee värillään, voidaanko työkalua käyttää kyseiseen kohtaan pelialueella. Mikäli työkalua voidaan käyttää, on osoitin vihreä. Muussa tapauksessa osoittimen väri on punainen.

ToolPointer tarkkailee myös törmäyksiä itsensä ja peliolioiden välillä. Kaikki työkaluja ei voida käyttää paikkoihin joissa sjaitsee jokin muu peliolio. Törmäystunnistukset suoritetaan yksinkertaisilla Flashista valmiiksi löytyvillä törmäystunnistusfunktioilla.

Käyttöliittymä sisältää lisäksi painikkeita joilla pelaaja voi aloittaa kentän uudestaa, pysäyttää pelin, hiljentää äänet tai palata päävalikkoon. Kaikkiin painikkeisiin on liitetty EventListenerejä jotka tarkkailevat pelaajan suorittamia hiiritoimintoja.

Peliin haluttiin toteuttaa myös peli-ikkunaa suurempia kenttiä joten siihen täytyi toteuttaa ruudun vieritysominaisuus. Vieritys toteutettiin siten, että luotiin ScrollHandler-luokka joka tarkkailee hiiren asemaa peli-ikkunassa. Mikäli hiiri viedään ikkunan vasempaan tai oikeaan reunaan ScrollHandler tarkastaa missä pelialua kyseisellä hetkellä sijaitsee. Jos pelialue on jo vieritettynä toiseen reunaan, ei sitä voida vierittää enempää. Mikäli tilaa vieritykselle on, muutetaan tarvittavien layereiden x-arvoa.

Pelin taustagrafiikoiden vierityksessä käytetään niin sanottua parallax scrollingia. Parallax scrolling tarkoittaa sitä, että tausta koostuu useammasta kerroksesta joita vieritetään eri nopeuksilla. Näin aikaansaadaan illuusia siitä, että jotkin taustan kerroksista ovat kauempana horisontissa kuin toiset. Mitä kaumpana tiettyä kerrosta halutaan esittää, sitä hitaammin sitä tulee vierittää suhteessa kerrokseen jossa kenttä ja pelihahmot sijaitsevat.

4.5 Roskien keruu

Pelin aikana luodaan valtava määrä erilaisia olioita. Jotta peli ei alkaisi vähitellen hidastua täytyy huolehtia roskien keruusta. Roskien keruu tapahtuu automaattisesti Actinscriptissä. Automaattinen roskien kerääjä ei kuitenkaan aina osaa päätellä milloin jokin olio voidaan tuhota joten sitä täytyy hieman avustaa.

Jotta olio voitaisiin merkata tuhottavaksi täytyy siitä poistaa kaikki EventListenerit ja poistaa se Display Lististä. Lisäksi täytyy pitää huoli siitä ettei mikään muu olio viittaa olioon joka halutaan tuhota. Olio voidaan tuhota vasta, kun kaikki viittaukset siihen on poistettu.

Tässä projektissa päätettiin toteuttaa jokaiselle luokalle RemoveSelf-funktio joka toimii ikään kuin hajottimena. Kaikissa luokissa joissa luodaan uusia oliota täytyy huolehtia siitä, että oliota tuhottaessa kutsutaan myös kaikkien luotujen olioiden RemoveSelf-funktiota joka on esitettynä listauksessa 8. Funktio-poistaa kaikki kysesen olion lapset, Event Listenerit ja poistaa olion display lististä. Parent oliossa viittaus kyseiseen olioon tulee vielä asettaa nulliksi.

```
public function removeSelf():void
{
    stageRef = null;
    objectLayer = null;

    removeEventListener(Event.ENTER_FRAME, onEnterFrame);
    parent.removeChild(this);
}
```

Listaus 8: Roskien keruuseen käytetty removeSelf-funktio

4.6 Valikot

Valikoita lähdettiin toteuttamaan, kun varsinainen peliosuus oli saatu valmiiksi. Peliin päätettiin toteuttaa kaksi suurempaa valikkoa: päävalikko ja kenttävalikko. Valikoihin haluttiin saada aikaiseksi hieman eloa joten niihin päätettiin toteuttaa vierivät taustagrafiikat.

Valikot muodostettiin Layer-luokan oliosta aivan kuten Game-luokkakin. Taaimmaisiin layereihin liitettiin taustagrafiikat ja ne asetettiin scrollaamaan automaattisesti halutulla nopeudella. Päävalikko koostuu vain muutamasta painikkeesta joilla pelaaja voi siirtyä kenttävalikkoon, pelin sponsorin kotisivuille tai pelin tekijän kotisivuille. Painikkeet toteutettiin EventListenerien avulla samaan tapaan kuin Game-luokan yhteydessä käytetyt valikot.

Kenttävalikkoa varten luotiin Selector-luokka jonka avulla voidaan valita haluttu kenttä. Selector-luokka on painikemainen olio joka ilmaisee kentän numeron ja onko kenttä auki vai lukittu. Selector olion rakennin on kuvattuna listauksessa 9.

```
public function Selector(X:int, Y:int, num:int)
{
    x = X;
    y = Y;
    selectorNumber = num;

    if(Main.savedLevel < selectorNumber)
    {
        gotoAndStop(3);
        locked = true;
    }
    else
    {
        numberText.text = String(selectorNumber);
        numberText.filters = [new GlowFilter(0x000000, 1, 4, 4, 10, 1,
           false, false)];
        gotoAndStop(1);
        addEventListener(MouseEvent.ROLL_OVER, onRollOver);
        addEventListener(MouseEvent.ROLL_OUT, onRollOut);
        addEventListener(MouseEvent.CLICK, onClick);
        locked = false;
    }
}
```

Listaus 9: Selector-luokan rakennin

Mikäli pelaaja ei ole vielä avannut kyseistä kenttää esitetään Selector-olion grafiikoissa lukon kuva eikä oliota voi klikata. Jos Selector-oliolle määritetty kenttä on jo avattu, voidaan sitä klikkaamalla siirtyä pelaamaan kyseistä kenttää. Selector-olio välittää kentän numeron Main-luokalle joka puolestaan vie sen Game-luokan rakentimeen jossa määritetään mistä kentästä peli alkaa.

Selector-oliot luodaan kenttävalikkoon kahdessa sisäkkäisessä for-loopissa. For-looppien avulla painikkeet saadaan järjesteltyä ruudulle siistiin järjestykseen. Jokaisella loopin iteraatiolla luodaan ruudulle yksi Selector-olio jonka sijainti x-akselilla määräytyy sisemmän loopin arvon mukaisesti ja y-koordinaatti ulomman loopin arvon perusteella.

5. Viimeistely

Tässä osiossa tutustutaan pelin viimeistelyn erivaiheisiin. Osiossa luodaan katsaus pelin äänien toteuttamiseen, graafisiin efekteihin ja esilataukseen.

5.1 Äänet

Äänien hyödyntäminen Flashissa on varsin yksinkertaista. Ääni-tiedostoja tuodaan kehitysympäristöön samalla tavalla kuin grafiikkaakin. Ääni-tiedostot toimivat ikään kuin luokat Actionscriptissä. Niille määritetään kehitysympäristössä nimi jonka avulla niitä voidaan hyödyntää koodin tasolla.

Jotta ääni voitaisiin toistaa, luodaan siitä uusi instanssi. Tämän jälkeen luodaan SoundChannel-luokan olio. SoundChannel-luokka on yksi Flashin valmiista luokista jonka avulla jokaiselle äänelle voidaan luoda oma kanavansa. Jokainen ääni tarvitseekin oman kanavansa, koska yhtä kanavaa käyttämällä voitaisiin toistaa ainoastaan yksi ääni kerrallaan.

Ääni liitetään kanavaan jonka jälkeen kutsutaan äänen play-funktiota. Play-funktioon voidaan viedä parametrina numeroarvo joka määrittää kuinka monta kertaa kyseinen ääni toistetaan. Tästä on hyötyä siinä vaiheessa, kun halutaan toistaa taustamusiikkeja. Taustamusiikkien tulee toistua jatkuvana silmukkana. Äänien käyttämiseen tarvittavaa koodia on kuvattu listauksessa 10.

```
public function playJumpSound():void
{
    if(!muted)
    {
        var sndJump:jump_sound = new jump_sound();
        var sndJumpChannel:SoundChannel;
        sndJumpChannel=sndJump.play();
    }
}
```

Listaus 10: Funktio joka luo uuden äänikanavan ja toistaa siihen liitetyn äänen

Tässä projektissa äänten hallintaan luotiin SoundHandler-luokka. SoundHandler-luokassa luodaan instanssit pelin musiikeista joita toistetaan valikoiden ja pelin aikana. Äänten toistamiseen on luotu jokaiselle äänelle omat funktionsa jotka luovat äänen ja kanavan ääntä varten. Ääniä tarvitsevat oliot voivat sitten vain yksinkertaisesti kutsua tarvittavaa funktiota SoundHandler-luokasta.

Äänten vaientamiseen on oma mute-funktio joka pysäyttää kaikki musiikit ja asettaa muted-arvon todeksi. Jokainen ääni funktio tarkistaa muted-arvon perusteella voidaanko ääni toistaa.

5.2 Graafiset efektit

Pelin viimeistely koostuu pääosin graafisesta ohjelmoinnista jossa pelistä pyritään tekemään elävämmän oloinen. Kaikkiin pelin painikkeisiin lisättiin ROLL_OVER- ja ROLL_OUT-EventListenerit joidenka avulla painikkeiden grafiikkaa voitiin muuttaa pelaajan viedessään hiiren osoittimen painikkeen päälle. Tällaiset pienet yksityiskohdat lisäävät tuovat peliin ammattilaismaisemman otteen ja lisäävät huomattavasti pelin arvoa.

Pelin tilojen välillä liikkumiseen luotiin FadeOutRect-luokka joka pimentää ruudun ennen kuin peli siirtyy tilasta toiseen. Kun ruutu on pimennetty, tuhotaan vanhat oliot ja luodaan uudet tilalle jonka jälkeen peliruudun peittävä FadeOutRect-luokan olio häivytetään näkyvistä. FadeOutRectin avulla pelitilojen välillä liikkuminen on huomattavasti sulavamman oloista.

5.3 Esilataus

Yksi tärkeimmistä viimeistelyn vaiheista on esilatauksen toteuttaminen. Koska tämän projektin peli on takoitettu toimimaan selaimella, täytyy pelaajan odottaa pelin latautumista palvelimelta. Flash ei voi esittää mitään sisältöä ennen kuin se on ladattu käyttäjän koneelle. Ilman esilatauksen toteuttamista pelaaja joutuisi katselemaan

valkoista ruutua niin kauan kunnes peli on täysin latautunut. Tästä johtuen peleihin on lähes välttämätöntä toteuttaa jonkinlainen latauspalkki joka ilmaisee pelin latautumisen tilan.

Tässä projektissa esilautaus toteutettiin luomalla Preloader-luokka jonka koodi ja grafiikat ladataan ennen muuta sisältöä. Preloader tarkkailee latauksen edistymistä ja esittää pelaajalle graafisen kuvauksen latauksen tilasta. Kun kaikki pelin sisältö on latautunut, Preloader luo ruudulla painikkeen jota painamalla pelaaja voi aloittaa pelin suorittamisen.

6. Yhteenveto

Työlle asetettu tavoite saatiin toteutettua ja kehityksen kohteena ollut peli saatiin valmiiksi. Työssä kehitettyä pelimoottoria voidaan jatkossa käyttää lähtökohtana uusien pelien kehittämiseen. Projektissa käytetyt Flash-työkalut toimivat moitteitta ja työn haasteet liittyivätkin lähinnä pelikehitykseen yleisellä tasolla eikä niinkään Flash-teknologian rajoituksiin.

Flash pelimarkkinat jatkavat kasvuaan ja Flashin tulevaisuus näyttää lupaavalta. Vaikka kilpailevia tekniikoita ilmestyy jatkuvasti, näyttää Flash pystyvän pitämään johtoasemansa selainpelimarkkinoilla. Internetissä on tuhansia peliportaaleja jotka ovat riippuvaisia Flash-peleistä joten Flash tulee pitämään valta-asemansa vielä pitkälle tulevaisuuteen. Flashin kehitys jatkuu kaiken aikaa ja mobiililaitteiden tehon kasvaessa se tulee varmastikin siirtymään tulevaisuudessa vahvemmin mobiilimaailmaan.

Flashilla voidaan toteuttaa nopeasti erilaisia pelimoottoreita jolloin aikaa jää enemmän pelin sisällön ja viimeistelyn toteuttamiseen. Flashin nopea kehitystyyli ruokkii myös luovuutta pelikehityksessä. Kaiken kaikkiaan Flash on erinomainen kehitysalusta joka tarjoaa monipuoliset työkalut pelikehitykseen.

Lähteet

Wikipedia: Adobe Flash[www-dokumentti]. [Viitattu 13.4.2011]
Saatavissa: http://fi.wikipedia.org/wiki/Adobe_Flash